JOURNAL OF COMPLEXITY **13,** 208–234 (1997)
ARTICLE NO. CM970442

The Fast Generalized Parker–Traub Algorithm for Inversion of Vandermonde and Related Matrices*

I. Gohberg†

School of Mathematical Sciences, Tel Aviv University, Ramat Aviv 69978, Israel

and

V. Olshevsky‡

Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, California 94305-4055

Received February 1996

In this paper we compare the numerical properties of the well-known *fast* $O(n^2)$ Traub and Björck–Pereyra algorithms, which both use the special structure of a Vandermonde matrix to rapidly compute the entries of its inverse. The results of numerical experiments suggest that the Parker variant of what we shall call the Parker–Traub algorithm allows one not only *fast* $O(n^2)$ inversion of a Vandermonde matrix, but it also gives more *accuracy*. We also show that the Parker–Traub algorithm is connected to the well-known concept of *displacement rank,* introduced by Kailath, Kung, and Morf about two decades ago, and therefore this algorithm can be generalized to invert the more general class of *Vandermonde-like* matrices, naturally suggested by the idea of displacement. © 1997 Academic Press

*This work was supported in part by ARO Contract DAAH04-96-1-0176, NSF Contract CCR-962811, and also by DARPA Contract F49620-95-1-0525. The views and conclusions contained in these documents are those of the author(s) and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the National Science Foundation, the Advanced Research Projects Agency, the Army Research office, or the U.S. Government.

†E-mail: gohberg@math.tau.ac.il.

‡E-mail: olshevsk@isl.stanford.edu; WWW: http://www-isl.stanford.edu/people/olshevsk.

0885-064X/97 $25.00

0. INTRODUCTION

We first consider the numerical inversion of Vandermonde matrices,

$$v(x) = \begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix}, \qquad \text{where } x = (x_i)_{i=0}^{n-1} \in \mathbf{C}^n. \tag{0.1}$$

Such matrices are ill-conditioned [5, 30], and standard numerically stable methods in general fail to compute the entries of their inverses accurately. The use of the structure of $V(x)$ may allow one to avoid the above difficulty and to achieve high relative accuracy. In this paper we compare two well-known methods that exploit the special structure in (0.1), the Traub and the Björck–Pereyra algorithms.

0.1. *The Traub Algorithm*

The algorithm, proposed by Traub in [29], is *fast* in the sense that it computes all n^2 entries of $V(x)^{-1}$ in only $6n^2$ floating point operations (flops), which compares favorably with the complexity $O(n^3)$ flops of general purpose algorithms. At the same time this algorithm is widely regarded as being numerically unstable, and the first indication of this can be found in [29].

0.2. *The Björck–Pereyra Algorithm*

In [1], Björck and Pereyra showed how to solve a Vandermonde linear system of the form

$$V(x) \cdot a = f, \qquad f \in \mathbf{C}^n \tag{0.2}$$

in only $5n^2/2$ flops. Clearly a Vandermonde matrix can be inverted by applying the Björck–Pereyra algorithm to solve n linear systems, using the columns of identity matrix for the right-hand sides. The latter $O(n^3)$ scheme is no longer fast. But, for the special case of positive and monotonically ordered points

$$0 < x_1 < x_2 < \cdots < x_n, \tag{0.3}$$

an error analysis of [15] implies the following favorable bound:

$$|\hat{V}(x)^{-1} - V(x)^{-1}| \leq 5nu|V(x)^{-1}| + O(u^2). \tag{0.4}$$

Here $\hat{V}(x)^{-1}$ stands for the inverse matrix computed by the Björck–Pereyra algorithm, u is the machine precision, and the operation of taking the absolute value and comparison of matrices are understood in a componentwise sense.

In [15] the pleasing bound (0.4) was used to argue that the fast $O(n^2)$ Traub algorithm is inaccurate, whereas the slow $O(n^3)$ Björck–Pereyra algorithm is the other way around. To the best of our knowledge the possibility of simultaneously fast and accurate inversion of a Vandermonde matrix was not reported anywhere. The results of our numerical experiments indicate that the algorithm, described next, satisfies these requirements; thus *one does not need to sacrifice the accuracy to achieve speed*.

0.3. *The Parker Algorithm*

The Traub inversion algorithm of [29] was several times independently derived in the engineering and mathematical literature, and several variants of this algorithm can be found in [4, 27, 31] and [18]. Interestingly, the Parker algorithm of [27] differs from the Traub algorithm only in one nonessential detail (see, *e.g.*, the main text below), and hence it is also subject to the comment noted above, implying that the result of the form (0.4) may not hold for the Parker variant as well. At the same time our numerical experiments show that the small difference between the Parker and the Traub algorithms is crucial, from the numerical point of view. Moreover, even in the situation most favorable for the Björck–Pereyra algorithm, *viz.* when (0.3) holds, the numerical performance of the Parker algorithm turned out to be not worse. In fact it is much better in the other cases, not captured by (0.3). *This occurrence reminds us that just comparing error bounds alone cannot be a reliable basis for making practical recommendations*.

The Parker inversion algorithm also allows one to solve a Vandermonde system (0.2) by forming $a = V(x)^{-1} \cdot f$, and our numerical experiments indicate that this method provides a very high relative accuracy in the computed solution. This occurrence contradicts a popular advice to avoid the use of the computed inverse in a larger computation, thus warning that *many generally valid guidelines cannot be automatically carried over to the problems where structured matrices are involved*.

0.4. *Displacement Structure*

The concept of *displacement structure* was first introduced in [20], and later it was much studied and generalized, see, *e.g.*, recent review [10] for historical remarks, list of applications and further references. This approach provides a unified method for the design of fast algorithms for various classes of structured matrices. In particular we shall show that the Parker–Traub algorithm can be derived by using the fact that Vandermonde matrices have displacement structure. Moreover, we shall show that this algorithm can be generalized to invert the wider class of *Vandermonde-like matrices*, naturally suggested by the concept of displacement.

0.5. *Contents*

The Traub, Parker, and the Björck–Pereyra algorithms are briefly reviewed in Sections 1–3, respectively. In Section 4 we compare these algorithms, observing that the Parker algorithm is also subject to the pessimistic comment of [15], noted above. Contrary to this expectation, the Parker algorithm demonstrated a very high accuracy in all our experiments, a small part of which is described in Sections 5 and 6. Finally, in Section 7 we reveal a connection between the Parker–Traub algorithm, and the concept of displacement structure. This association allowed us to carry over in Section 8 this fast inversion algorithm to the wider class of Vandermonde-like matrices.

It is our pleasure to thank Thomas Kailath for useful remarks.

1. THE TRAUB ALGORITHM

In [29] Traub proposed a fast method to compute all n^2 entries of $V(x)^{-1}$ in only $6n^2$ flops. Because of the importance for the further analysis and generalizations, we shall briefly review the Traub algorithm, which is based on the use of an explicit formula for $V(x)^{-1}$. Let

$$P(x) = \prod_{k=1}^{n} (x - x_k) = x^n + \sum_{k=0}^{n-1} a_k \cdot x^k \tag{1.1}$$

be the *master* polynomial, whose zeros are the nodes of $V(x)$. Following [29], consider the divided difference

$$P[t, x] = \frac{P(t) - P(x)}{t - x}, \tag{1.2}$$

and define the polynomials

$$\{q_k(x)\}_{0 \le k \le n} \qquad \text{with } q_n(x) = P(x) \tag{1.3}$$

by

$$P[t, x] = q_{n-1}(x) + t \cdot q_{n-2}(x) + \cdots + t^{n-2} \cdot q_1(x) + t^{n-1} \cdot q_0(x). \tag{1.4}$$

The polynomials in (1.3) are called the *associated polynomials* of $P(x)$, or sometimes the *Horner polynomials*, see, *e.g.*, [29] for historical remarks. Substituting (1.1) into (1.2), one sees that the bivariate function $P[t, x]$ has a Hankel structure

$$P[t, x] = \sum_{i=1}^{n} a_i \cdot \frac{t^i - x^i}{t - x} = \sum_{i=1}^{n} a_i \cdot (t^{i-1} + t^{i-2}x + \cdots + tx^{i-2} + x^{i-1}),$$

which implies that the associated polynomials are given by

$$q_k(x) = x^k + a_{n-1} \cdot x^{k-1} + \cdots + a_{n-k+1} \cdot x + a_{n-k}. \tag{1.5}$$

Equivalently, they satisfy the recurrence relations

$$q_0(x) = 1, \qquad q_k(x) = x \cdot q_{k-1}(x) + a_{n-k} \ (k = 1, 2, \cdots, n). \tag{1.6}$$

From (1.1), (1.2), (1.4), and the trivial identity

$$P[x, x] = P'(x), \tag{1.7}$$

we obtain what Traub called the *basic orthonormality relation*:

$$\frac{P[x_j, x_k]}{P'(x_k)} = \sum_{i=1}^{n-1} x_j^i \cdot \frac{q_{n-1-i}(x_k)}{P'(x_k)} = \delta_{jk}.$$

The latter relation in turn implies that the inverse of a Vandermonde matrix (0.1) is given by

$$V^{-1}(x) = \begin{bmatrix} q_{n-1}(x_1) & q_{n-1}(x_2) & \cdots & q_{n-1}(x_n) \\ q_{n-2}(x_1) & q_{n-2}(x_2) & \cdots & q_{n-2}(x_n) \\ \vdots & \vdots & & \vdots \\ q_0(x_1) & q_0(x_2) & \cdots & q_0(x_n) \end{bmatrix} \cdot \operatorname{diag}\left(\left[\frac{1}{P'(x_i)} \right]_{i=1}^{n} \right). \tag{1.8}$$

Traub exploited formula (1.8) to derive his fast algorithm for inversion of Vandermonde matrices. To describe his procedure, let us observe that (1.6) allows one to compute the entries of the first factor in (1.8), whereas the entries $P'(x) = q_n'(x)$ of the second factor can be computed by

$$q_1'(x) = a_{n-1}, \qquad q_k'(x) = q_{k-1}(x) + x \cdot q_{k-1}'(x) \qquad (k = 2, 3, \cdots, n), \tag{1.9}$$

which are obtained by differentiation of (1.6). Thus the Traub algorithm can be summarized as follows.

THE TRAUB ALGORITHM.

(1) Compute the coefficients of $P(x)$ in (1.1) via nested polynomial multiplication:

$$\begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \end{bmatrix} = \begin{bmatrix} -x_1 \\ 1 \end{bmatrix}, \qquad \begin{bmatrix} a_0^{(k)} \\ a_1^{(k)} \\ \vdots \\ a_k^{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ a_0^{(k-1)} \\ \vdots \\ a_{k-1}^{(k-1)} \end{bmatrix} - x_k \cdot \begin{bmatrix} a_0^{(k-1)} \\ \vdots \\ a_{k-1}^{(k-1)} \\ 0 \end{bmatrix}, \quad (1.10)$$

with $a_j = a_j^{(n)}$.

(2) For $j = 1, 2, \cdots, n$ do:
   (a) Compute $q_k(x_j)$ ($k = 0, 1, \cdots, n-1$) via (1.6).
   (b) Using these quantities compute $P'(x_j) = q'_n(x_j)$ by (1.9).
   (c) Compute the jth column $[q_k(x_j)/P'(x_j)]_{0 \le k \le n-1}$ of $V^{-1}(x)$.

The Traub algorithm computes all n^2 entries of $V(x)$ in only $6n^2$ flops, which compares favorably with the complexity $O(n^3)$ flops of standard (structure-ignoring) methods. However, as was stated, this algorithm is subject to the rapid propagation of roundoff errors, and, as a matter of fact, it produces inaccurate solutions. As we shall see in Sections 5 and 6, a fast algorithm described next exhibits much better numerical properties.

2. THE PARKER ALGORITHM

We discovered that earlier than [29], Parker described in [27] an inversion procedure, which is based on another variant of the inversion formula described below.

For $j = 1, 2, \cdots, n$ denote by $L_j(x)$ the jth Lagrange polynomial, i.e. the unique polynomial of degree $n-1$ satisfying

$$L_j(x_j) = 1, \qquad L_j(x_k) = 0 \quad (k \neq j). \quad (2.1)$$

It follows immediately from (2.1) that the columns of the inverse of a Vandermonde matrix in (0.1) are formed from the coefficients of $L_j(x) = l_{n-1,\,j} \cdot x^{n-1} + l_{n-2,\,j} \cdot x^{n-2} + \cdots + l_{1,\,j} \cdot x + l_{0,\,j}$, i.e.

$$V^{-1}(x) = \begin{bmatrix} l_{0,1} & l_{0,2} & \cdots & l_{0,n} \\ l_{1,1} & l_{1,2} & \cdots & l_{1,n} \\ \vdots & \vdots & & \vdots \\ l_{n-1,1} & l_{n-1,2} & \cdots & l_{n-1,n} \end{bmatrix}. \quad (2.2)$$

The description (2.2) can be found in [19], and later it was often rediscovered by many authors, see, *e.g.,* the remarks in [29]. In particular, Parker rederived in [27] the formula (2.2) and, anticipating the Traub algorithm, observed that it

suggests an efficient inversion procedure for the Vandermonde matrix. Parker's arguments are based on the following well-known identities. Let $P(x)$ be the master polynomial, defined in (1.1), then the jth Lagrange polynomial is given by

$$L_j(x) = \frac{P(x)}{(x - x_j) \cdot P'(x_j)}, \tag{2.3}$$

where clearly

$$\begin{aligned} P'(x_j) = &(x_j - x_1) \cdot \ldots \cdot (x_j - x_{j-1}) \\ &\cdot (x_j - x_{j+1}) \cdot \ldots \cdot (x_j - x_n). \end{aligned} \tag{2.4}$$

In fact Parker outlined the following scheme. First one computes the coefficients of the master polynomial $P(x) = \sum_{i=0}^{n} a_i x^i$, then divides it synthetically by $(x - x_j)$. It is easy to see that the synthetic division gives rise to the following recursion

$$q_{0,\,j} = 1, \qquad q_{k,\,j} = x_j \cdot q_{k-1,\,j} + a_{n-k}. \tag{2.5}$$

for the coefficients of the quotient

$$\frac{P(x)}{(x - x_j)} = \sum_{k=0}^{n-1} q_{k,\,j} x^{n-1-k}. \tag{2.6}$$

To finally obtain the coefficients of $L_j(x)$ in (2.3), or equivalently the entries of the jth column of $V^{-1}(x)$ in (2.2), it remains only to divide the polynomial in (2.6) by $P'(x_j)$, the latter is computed via (2.4). Thus the inversion algorithm can be summarized as follows.

THE PARKER ALGORITHM.

(1) Compute the coefficients of $P(x)$ in (1.1) by (1.10).
(2) For $j = 1, 2, \cdots, n$ do
    (a) Compute $q_{k,\,j}$ $(j = 0, 1, \cdots, n - 1)$ by recursion (2.5).
    (b) Compute $P'(x_j)$ by (2.4).
    (c) Compute the jth column $[q_{k,\,j}/P'(x_j)]_{0 \le k \le n-1}$ of $V^{-1}(x)$.

The computational complexity $6n^2$ flops of the above algorithm[1] was not counted by Parker, who however indicated that his algorithm is fast, by noting

[1]By a proper implementation of the step 2.b, this complexity can be further reduced to $5.5n^2$ flops, see, *e.g.*, [11], where computing $P'(x_j)$ appears as a part of a fast $O(n^2)$ algorithm for inversion of Chebyshev–Vandermonde matrices.

that "*pencil and paper calculation of the inverse of a matrix of order six takes about twenty minutes.*"

A direct comparison reveals that the only difference between the Traub and the Parker schemes is in the step 2.b, thus making clear that they are just different versions of the same algorithm. Interestingly, due to the importance for applications, the fast Parker–Traub algorithm was often rederived in the engineering literature, see, for example, [4, 31] [2], and [18].

We shall demonstrate by computed examples in Sections 5, 6 that the Parker variant of inversion algorithm demonstrates in practice a very satisfactory numerical performance. However before doing so, we review another popular inversion procedure, whose numerical properties were much analyzed in the numerical analysis literature.

3. THE BJÖRCK–PEREYRA ALGORITHM

Björck and Pereyra described in [1] a fast algorithm that solves a Vandermonde linear system

$$V(x) \cdot a = f, \qquad \text{with } a = [a_k]_{1 \le k \le n},\ f = [f_k]_{1 \le k \le n}. \tag{3.1}$$

in only $5n^2/2$ flops. Their algorithm exploits another explicit expression for $V(x)^{-1}$, stated next.

$$V(x)^{-1} = U_1 \cdot U_2 \cdots U_{n-1} \cdot D_{n-1} \cdot L_{n-1} \cdot D_1 \cdot L_1, \tag{3.2}$$

where

$$U_i = \left[\begin{array}{c|ccccc} I_{i-1} & & & & & \\ \hline & 1 & -x_i & & & \\ & & 1 & & & \\ & & & \ddots & -x_i & \\ & & & & 1 & \end{array}\right]$$

and

[2]The important problem of decoding of the widely used Reed–Solomon codes is naturally divided into two parts: (a) determining error locations (which can be done by the well-known Berlecamp–Massey algorithm), and then (b) computing the actual errors. The second problem is equivalent to solving a Vandermonde linear system, which can be done by the Forney algorithm (see, *e.g.,* [4], p. 119). In fact, the Forney algorithm is yet another variant of the Parker–Traub algorithm, and again the only difference from the other two is in the step 2.b. But the numerical performance of the Forney algorithm is not particularly important in the context of the coding theory, because all the computation is done over $GF(q)$, so there are no roundoffs.

$$D_i = \left[\begin{array}{c|ccccc} I_i & & & & & \\ \hline & \frac{1}{x_{i+1}-x_1} & & & & \\ & & \frac{1}{x_{i+2}-x_2} & & & \\ & & & \ddots & & \\ & & & & \frac{1}{x_n - x_{n-1}} \end{array}\right],$$

$$L_i = \left[\begin{array}{c|cccc} I_{i-1} & & & & \\ \hline & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{array}\right]$$

This formula yields the following algorithm for solving a linear system in (3.1):

THE BJÖRCK–PEREYRA ALGORITHM.

for $k = 1, 2, \cdots, n$ **do**
  $a_k = f_k$
for $k = 1, 2, \cdots, n-1$ **do**
  for $i = n, n-1, \cdots, k+1$ **do**

$$a_i = \frac{a_i - a_{i-1}}{x_i - x_{i-k-1}} \tag{3.3}$$

  endfor
endfor
for $k = n-1, n-2, \cdots, 1$ **do**
  for $i = k, k+1, \cdots, n-1$ **do**

$$a_i = a_i - a_{i+1} \cdot x_k$$

  endfor
endfor

This algorithm solves one linear Vandermonde system of the form (3.1) in only $5n^2/2$ flops. Clearly it can be used for computing the entries of $V(x)^{-1}$ by solving n linear systems, using the columns of identity matrix for the right-hand sides. The latter inversion scheme is no longer fast, and it requires performing $O(n^3)$ flops, thus loosing a superiority in speed over standard (structure-ignoring) algorithms. But it is known to provide, for special configurations of the points $\{x_k\}$, much more accurate results than standard numerically stable algorithms, as reviewed in the next section.

4. COMPARISON OF THE PARKER–TRAUB AND THE BJÖRCK–PEREYRA INVERSION ALGORITHMS

Björck and Pereyra observed in [1] that their algorithm frequently produces more accurate solutions than could be expected from the condition number of the coefficient matrix. In [15] Higham analyzed Vandermonde matrices with positive and monotonically ordered points,

$$0 < x_1 < x_2 < \cdots < x_n, \tag{4.1}$$

and showed that (for this specific subclass) the Björck–Pereyra algorithm computes the inverse matrix $\hat{V}(x)^{-1}$ so that the error is as small as could possibly be expected:

$$|\hat{V}(x)^{-1} - V(x)^{-1}| \leq 5nu \cdot |V(x)^{-1}| + O(u^2). \tag{4.2}$$

Here u is the machine precision, and the comparison and the operation of taking the absolute value of a matrix, are understood in a componentwise sense.

The latter surprisingly pleasing bound was used in [15] to compare the Traub and the Björck–Pereyra algorithms, and it was pointed out there that (4.2) "*shows that contrary to what one might expect, $V(x)^{-1}$ can be computed with high relative accuracy*." Then [15] turned to the Traub algorithm: "*However [29] does not contain a rounding error analysis, and it can be shown that Traub's $O(n^2)$ algorithm must involve subtraction of like-signed numbers, suggesting that a result of the form (4.2) will not hold.*"

Let us now recall that the Traub and the Parker schemes are essentially the same algorithm (the only difference between them is in the step 2.b, see, *e.g.*, Sections 1, 2). Therefore the Parker variant also involves the subtraction of like-signed numbers.[3] Thus, the Parker variant of the Parker–Traub algorithm is also subject to the above remark, saying that a result of the form (4.2) will not hold for the Parker algorithm as well. Summarizing, the arguments of [15] suggest that the slow $O(n^3)$ Björck–Pereyra algorithm is accurate, whereas the fast $O(n^2)$ Parker–Traub inversion algorithm is the other way around.[4] At the same time the results of our numerical experiments, only small part of which is described in the next section, show that the Parker version (combined with

[3]Interestingly, if the condition (4.1) holds, then *each step* 2.a (since the recursions (1.6) and (2.5) coincide, this step is the same for both algorithms) must contain such a subtraction. Indeed, in the case of (4.1), $V(x)$ is known to be a totally positive matrix, see, *e.g.*, [6]. Recall that totally positive matrices are defined as those whose all minors are positive, and hence the columns of their inverses have the sign-oscillation property. Applying this to $V(x)^{-1}$ in (1.8), one sees that we have $q_k(x_j) \cdot q_{k-1}(x_j) < 0$, so that by (1.6) *each step* 2.a of both the Parker and the Traub algorithms must involve the subtraction of like-signed numbers.

[4]Moreover, to the best of our knowledge, the possibility of simultaneously fast and accurate inversion of $V(x)$ was not reported anywhere.

an appropriate reordering of the points $\{x_k\}$) turns out to be very reliable in practice. Thus it is possible to speed-up the computation without sacrificing the accuracy. Moreover, this occurrence reminds us that just comparing error bounds *alone* should not be a basis for making reliable recommendations.

5. NUMERICAL EXPERIMENTS WITH INVERSION

In this section we present several fairly representative numerical examples, comparing the accuracy of the following algorithms:

(1)	GJECP	$O(n^3)$	Gauss–Jordan elimination with complete pivoting.
(2)	B-P	$O(n^3)$	The Björck–Pereyra algoirthm, applied to solving n linear systems, using the columns of identity matrix for the right-hand sides.
(3)	Traub	$O(n^2)$	The Traub inversion algorithm.
(4)	Parker	$O(n^2)$	The Parker inversion algorithm.

5.1. *Accuracy*

All the above algorithms were implemented on a DEC workstation in both single and double precision, for which the unit roundoffs are $u_s = 2^{-23} \approx 1.19 \times 10^{-7}$, and $u_d \approx 2^{-56} = 1.4 \times 10^{-17}$, respectively. To check the accuracy we accomplished the following procedure. Among four inverse matrices, computed in double precision by the above listed algorithms, we chosen two, say H_{d1} and H_{d2}, which minimized, in this particular example, the double precision relative error

$$e_d = \frac{||H_{d1} - H_{d2}||_2}{||H_{d1}||_2}.$$

If this number e_d (included in the Tables below) was sufficiently small, then we considered H_{d1} to be the exact solution, and used it to compute the single precision relative errors

$$e_s = \frac{||H_{d1} - H_{si}||_2}{||H_{d1}||_2},$$

for matrices H_{si} ($i = 1, 2, \cdots, 5$), computed in single precision by each of the five compared algorithms.

5.2. *Different Orderings of the Points*

It is our experience and the experience of many others, that the numerical behavior of many algorithms, related to polynomial and rational interpolation problems, depends upon the ordering of interpolation points. We considered the following three orderings of x_k.

- *Random ordering*.
- *Monotonic ordering*. The points are ordered so that $x_1 < x_2 < \cdots < x_n$.
- *Leja ordering*. The points x_i were reordered so that

$$|x_1| = \max_{1\le k\le n} |x_k|,$$

and

$$\prod_{j=1}^{k-1} |x_k - x_j| = \max_{k\le l\le n} \prod_{j=1}^{k-1} |x_l - x_j|, \qquad (2 \le k \le n); \tag{5.1}$$

the latter ordering is related to Leja work on interpolation, see, *e.g.*, [28] and references therein. In [16] Higham showed how to reorder x_k so that (5.1) holds, using only n^2 flops; thus incorporating the Leja ordering will not slow down any fast $O(n^2)$ algorithm. Moreover, it was shown in [16] that (5.1) mimics a row permutation of $V(x)$, which would be obtained by applying to $V(x)$ of *partial pivoting* technique. This fact will be used in Section 7, where we shall extend the Parker–Traub algorithm with partial pivoting to invert the more general class of Vandermonde-like matrices.

Clearly, the Gauss–Jordan algorithm with complete pivoting is numerically insensitive to reordering of x_k, and it turned out that the Traub algorithm also did not reveal in computed examples any correlation with different orderings. At the same time the accuracy of the Björck–Pereyra, and of the Parker algorithm do depend on ordering of x_k.

5.3. *Positive Points*

EXAMPLE 1 (Positive Equidistant Points). Table I presents the results of numerical inversion of $V(x)$, where x_k are the equidistant in the interval (0, 1) points.

The matrix in Example 1 becomes extremely ill-conditioned already for n = 10, so it is not surprising that the Gauss–Jordan elimination failed. Furthermore, as Table I shows, the Traub variant of the inversion algorithm also breaks down numerically. One sees that the nonessential difference in step 2.b of the Traub and the Parker algorithms turns out to be crucial from the numerical point of view. More precisely, the Traub algorithm computes the values $P'(x_j)$ via (1.9),

which involves subtraction of the *computed quantities* $q_{k-1}(x_j)$, and hence it is subject to the rapid propagation of roundoff errors. On the other hand, step 2.b of the Parker algorithm is based on the formula (2.4), which involves subtraction of only *input data*, and hence the Parker algorithm provides much more accurate solutions.

Recall that since $x_k > 0$, the Björck–Pereyra algorithm combined with monotonic ordering (4.1) is guaranteed to provide in Example 1 an exceptionally high accuracy, see, *e.g.*, (4.2). At the same time, Table I demonstrates that even in this most favorable for the Björck–Pereyra algorithm situation, the accuracy of the Parker algorithm is at least not worse. In fact it is much better in the other cases, as will be seen in the next examples.

5.4. *Points with Both Signs*

For this case there are no sharp stability results, like in (4.2), and moreover, experiments indicate that the Björck–Pereyra algorithm becomes less accurate, when the points x_k are of both signs. The next two tables demonstrate that this is not the case with the Parker algorithm, combined with Leja ordering.

EXAMPLE 2 (Equidistant in $(-1, 1)$ Points). See Table II.

Table I shows that if the points x_k are positive, then the Parker algorithm performs equally well for all three orderings. Table II demonstrates that if x_k are of both signs, the accuracy of the Parker algorithm breaks down with monotonic ordering. Moreover, in this case Leja ordering improves the numerical performance of both the Björck–Pereyra and Parker algorithms, and that the latter is the most accurate among compared algorithms. The numerical supremacy of the Parker algorithm over the Björck–Pereyra algorithm becomes even more appreciable in the next example.

TABLE I
Equidistant in (0, 1) Points

					Random ordering		Monotonic ordering		Leja ordering	
n	$\kappa_2(V)$	e_d	GJECP e_s	Traub e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s
5	2e+03	1e-16	8e-07	9e-06	4e-08	1e-07	**9e-08**	1e-07	5e-08	**1e-07**
10	6e+07	3e-16	3e-02	1e-01	2e-07	2e-07	**2e-07**	2e-07	3e-07	**2e-07**
20	4e+16	7e-16	1e+00	1e+00	2e-06	4e-07	**5e-07**	4e-07	5e-07	**5e-07**
30	4e+18	3e-12	1e+00	1e+00	6e-04	5e-07	**6e-07**	5e-07	3e-06	**6e-07**
40	8e+18	5e-13	1e+00	1e+00	3e-04	8e-07	**7e-07**	7e-07	7e-06	**7e-07**
50	6e-18	1e-11	1e+00	1e+00	Inf	Inf	Inf	Inf	Inf	Inf

TABLE II
Equidistant in (–1, 1) Points

					Random ordering		Monotonic ordering		Leja ordering	
n	$\kappa_2(V)$	e_d	GJECP e_s	Traub e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s
5	2e+01	4e-17	1e-07	2e-08	3e-08	2e-08	3e-08	2e-08	**4e-08**	**2e-08**
10	5e+03	4e-16	9e-06	3e-06	3e-07	3e-07	5e-07	1e-07	**4e-07**	**3e-07**
20	3e+08	3e-15	4e-01	9e-05	1e-06	3e-07	4e-06	4e-06	**3e-07**	**2e-07**
30	2e+13	2e-12	1e+00	4e-03	1e-03	3e-07	4e-05	7e-05	**5e-06**	**3e-07**
40	1e+18	4e-13	1e+00	1e-02	3e-04	8e-07	8e-04	1e-03	**6e-05**	**9e-07**
50	7e+18	5e-12	1e+00	2e-01	4e-03	2e-07	1e-02	7e-03	**2e-04**	**4e-07**
60	3e+19	1e-13	1e+00	2e-01	8e-05	4e-07	8e-02	1e-01	**2e-03**	**4e-07**

EXAMPLE 3 (Chebyshev Zeros in $(-1, 1)$). Table III contains the results for inversion of $V(x)$, where $x_i = \cos((2i-1)\cdot\pi/2\cdot n)$.

Examples 1–3 are fairly representative, and a careful search did not reveal any case where the Björck–Pereyra algorithm was more accurate than the Parker algorithm, whereas the latter demonstrated, in many examples, a strong numerical superiority over the Björck–Pereyra algorithm.

Finally note that recently the Traub algorithm was generalized to invert *Chebyshev–Vandermonde* matrices in [11], and what we call *three-term* Vandermonde

TABLE III
Chebyshev Zeros in (–1, 1)

					Random ordering		Monotonic ordering		Leja ordering	
n	$\kappa_2(V)$	e_d	GJECP e_s	Traub e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s
5	2e+01	3e-16	2e-07	3e-07	8e-08	8e-08	1e-07	5e-08	**2e-07**	**6e-08**
10	1e+03	6e-16	2e–06	5e-06	3e-07	1e-07	5e-07	5e-07	**5e-07**	**1e-07**
20	9e+06	3e-14	5e-02	2e–02	8e-06	2e-07	1e-05	1e-05	**2e-05**	**3e-07**
30	6e+10	3e-11	1e+00	3e–01	9e-03	3e-07	2e-04	7e-05	**5e-05**	**3e-07**
40	4e+14	2e-09	1e+00	5e–01	1e+00	4e-07	3e-03	6e-03	**2e-03**	**3e-07**
50	6e+17	4e-10	1e+00	6e–01	1e-01	6e-07	5e-02	2e-02	**1e-02**	**6e-07**
60	7e+18	9e-08	1e+00	7e–01	2e+01	6e-07	8e-01	1e+00	**5e-02**	**6e-07**

matrices in [2]. Now it is clear that both these algorithms are generalizations of the Parker rather than the Traub scheme. Favorable stability results were reported in both cases, whereas their prototype, i.e. the Traub algorithm, was generally regarded as being numerically unstable. We hope that the present paper explains this curious situation.

6. NUMERICAL EXPERIMENTS WITH VANDERMONDE SYSTEMS

The high relative accuracy shown by the Parker algorithm in Examples 1–3 suggests solving linear system $V(x)\cdot a = f$ by forming $V(x)^{-1}\cdot f$. This would be useful, for example, in the case of multiple right-hand sides, but numerous sources predict a numerical breakdown, thus ruling out such a possibility. For example [3], motivated by the question of what inversion methods should be used in LAPACK, concludes with the remark: "*we wish to stress that all the analysis here pertains to matrix inversion alone. It is usually the case that when a computed inverse is used as a part of a larger computation, the stability properties are less favorable, and this is one reason why matrix inversion is generally discouraged*." The paper [3] justifies this conclusion with a particular example where solving linear system $L\cdot a = f$ by forward substitution was more accurate than when using the inversion of a triangular matrix L.

Contrary to the above caution, the next two examples, as well as the results of other numerical experiments, indicate that the use of the Parker inversion algorithm provides a very high relative accuracy in the computed solution of $V(x)\cdot a = f$, showing strong numerical superiority over other compared algorithms.

EXAMPLE 4 (Points: Chebyshev Zeros in (0, 1), RHS ± 1). In this example we solved a linear system

$$V(x)\cdot a = f, \quad \text{where } x_i = \cos\left(\frac{(2i-1)\cdot\pi}{4\cdot n}\right),\ f_i = (-1)^i.$$

This is the most favorable for the Björck–Pereyra algorithm case of positive, monotonically ordered points x_k and sign-interchanging right-hand side, and as shown in [15], the Björck–Pereyra algorithm is guaranteed to compute in this case a remarkably accurate solution $\hat{a}$:

$$|\hat{a} - a| \le 5nu\cdot|a| + O(u^2). \tag{6.1}$$

Table IV demonstrates that the accuracy of the Björck–Pereyra algorithm, combined with monotonic ordering, is indeed compatible with the remarkable bound in (6.1). But even in this most pleasing for the Björck–Pereyra algorithm

TABLE IV
Points: Chebyshev Zeros in (0, 1), RHS ±1

						Random ordering		Monotonic ordering		Leja ordering	
n	$\kappa_2(V)$	$\|\|a\|\|_2$	e_d	GJECP e_s	Traub e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s
5	3e+03	2e+03	4e-16	6e-06	2e-05	2e-07	2e-07	**2e-07**	2e-07	3e-07	**2e-07**
10	2e+08	8e+07	2e-15	3e+00	9e-01	7e-08	2e-07	**4e-07**	3e-07	3e-07	**2e-07**
20	6e+17	4e+17	1e-15	1e+00	1e+00	4e-06	9e-07	**9e-07**	1e-06	4e-06	**1e-06**
30	6e+18	3e+27	5e-14	1e+00	1e+00	2e-04	7e-07	**7e-07**	7e-07	2e-05	**7e-07**
40	4e+18	3e+37	4e-12	1e+00	1e+00	5e-04	1e-06	**2e-06**	1e-06	5e-03	**1e-06**
45	7e+18	3e+42	4e-11	1e+00	1e+00	Inf	Inf	Inf	Inf	Inf	Inf

situation, the stability properties of the Parker algorithm are not worse. In fact they are better when the points x_k are of both signs, and they are either in Leja or random ordering, as illustrated by the next example.

EXAMPLE 5 (Points: Clustered in (−1, 1), RHS: Random in (0, 10)). Table V contains the results for solving linear system

$$V(x) \cdot a = f, \qquad \text{where } x_i = -1 + 2 \cdot \frac{i^2}{n^2},$$

and the components of the right-hand side f are random numbers in the interval (0, 10).

TABLE V
Points: Clustered in (–1, 1), RHS: Random in (0, 10)

						Random ordering		Monotonic ordering		Leja ordering	
n	$\kappa_2(V)$	$\|\|a\|\|_2$	e_d	GJECP e_s	Traub e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s	B-P e_s	Parker e_s
5	7e+01	5e+01	2e-16	2e-07	4e-08	**1e-07**	4e-08	2e-07	4e-08	1e-07	**2e-08**
10	1e+05	8e+04	5e-16	3e–04	2e-03	**2e-07**	3e-07	3e-07	3e-06	8e-07	**1e-07**
20	4e+11	2e+11	2e-15	1e+00	1e+00	**2e-06**	3e-06	4e-06	1e-05	7e-07	**3e-06**
30	2e+17	2e+17	2e-15	1e+00	1e+00	**4e-05**	8e-06	7e-05	7e-04	3e-04	**6e-06**
40	1e+18	2e+22	4e-13	1e+00	1e+00	**4e-01**	2e-04	3e-03	3e-04	1e-01	**2e-04**
50	2e+18	4e+30	7e-12	1e+00	1e+00	**7e-03**	2e-06	5e-03	3e-02	9e-02	**1e-06**
60	7e+18	5e+37	2e-13	1e+00	1e+00	**7e-05**	5e-07	2e-01	3e-01	8e-02	**1e-06**

The above two examples demonstrate that the above conclusion of [3] (as well as some other generally valid guidelines) cannot be automatically carried over to the special situations, in which structured matrices are involved.

7. VANDERMONDE-LIKE MATRICES

It turns out that not just Vandermonde, but the more general class of Vandermonde-like matrices, defined below, can be inverted in $O(n^2)$ arithmetic operations. Our goal in the rest of the paper is to formulate the *generalized Parker–Traub* algorithm for this purpose.

7.1. *The Structure of the Inverse of a Vandermonde Matrix*

Recall that the Parker–Traub algorithm is based on formula (1.8), which in view of (1.5) can be immediately rewritten as follows

$$V^{-1}(x) = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_{n-1} & 1 \\ a_2 & a_3 & \cdots & a_{n-1} & 1 & 0 \\ a_3 & & \iddots & \iddots & & \vdots \\ \vdots & \iddots & \iddots & & & \vdots \\ a_{n-1} & \iddots & & & & \vdots \\ 1 & 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \cdot V(x)^T \cdot \operatorname{diag}\left(\left[\frac{1}{P'(x_i)} \right]_{i=1}^{n} \right). \tag{7.1}$$

More specifically, the Parker–Traub algorithm inverts $V(x)$ in two basic steps:

(PT1) Compute the entries of the matrices on the right-hand side of (7.1). This is done in the steps 1 and 2.b of the Parker–Traub algorithm.

(PT2) Compute the product of a Hankel, transpose Vandermonde and a diagonal matrices on the right-hand side of (7.1). This is done in the steps 2.a and 2.c of the Parker–Traub algorithm.

Moreover, the Parker–Traub algorithm achieves a favorable complexity $O(n^2)$ flops by exploiting the following properties of a Vandermonde matrix.

(A) $V(x)^{-1}$ has the special structure shown in (7.1), i.e. it is a product of a Hankel, transposed Vandermonde and diagonal matrices.

(B) The entries of the matrices on the right-hand side of (7.1) can be computed in $O(n^2)$ flops.

(C) The product of the Hankel and transposed Vandermonde matrices on the right-hand side of (7.1) can be computed in $O(n^2)$ flops using the recursion in (1.6).

Moreover, the Parker algorithm achieves a high relative accuracy by incorporation of Leja ordering, i.e. by exploiting next property.

(D) Partial pivoting technique can be incorporated without increasing the $O(n^2)$ complexity of the algorithm.

It turns out that the above properties reflect the fact that $V(x)$ has *displacement structure*. Therefore the Parker–Traub algorithm can be extended to invert more general *Vandermonde-like* matrices, naturally suggested by the concept of displacement.

7.2. *Displacement Structure*

The displacement structure theory was started by T. Kailath, Kung, and Morf in [20], where it was first applied to the study of Toeplitz matrices. They considered a *displacement operator* $\nabla_{\{Z_0,\, Z_0\}}(\cdot) : \mathbf{C}^{n\times n} \to \mathbf{C}^{n\times n}$ of the form

$$\nabla_{\{Z_0,\, Z_0\}}(R) = R - Z_0 \cdot R \cdot Z_0^T, \tag{7.2}$$

where Z_0 is the lower shift matrix, i.e. with ones on the first subdiagonal and zeros elsewhere; $Z_0 \cdot R \cdot Z_0^T$ then corresponds to shifting R downwards along main diagonal. The matrix $\nabla_{\{Z_0,\, Z_0\}}(R)$ was called $\{Z_0,\, Z_0\}$-*displacement* of R, and the number rank$\nabla_{\{Z_0,\, Z_0\}}(R)$ was called $\{Z_0,\, Z_0\}$-*displacement rank* of R. Kailath et al. observed that the shift-invariance property of Toeplitz matrices implies that their $\{Z_0,\, Z_0\}$-displacement rank do not exceed 2, and showed that many properties of ordinary Toeplitz matrices are naturally carried over to more general *Toeplitz-like* matrices, defined as those with low $\{Z_0,\, Z_0\}$-displacement rank. Later it has been realized (starting from [17]) that the concept of displacement suggests a unified approach to the study of Hankel, Toeplitz-plus-Hankel, Vandermonde, Chebyshev-Vandermonde, Cauchy, Pick matrices, Bezoutians, and many other classes of structured matrices, see, *e.g.*, recent review [10] for historical remarks, list of applications and further references. In particular Heinig and Rost observed in [17] that Vandermonde matrices are transformed to rank-one matrices by a suitable displacement operator, and used this fact to carry over many results to more general *Vandermonde-like* matrices, whose formal definition can be found in the next subsection. In the rest of the paper we use the displacement structure approach to show that the above properties (A)–(D) hold not just for $V(x)$, but also for Vandermonde-like matrices. This will allow us to obtain fast $O(n^2)$ *generalized Parker–Traub algorithm* with *partial pivoting* for inversion of Vandermonde-like matrices.

7.3. *Vandermonde-like Displacement Structure*

For our purposes here it will be more convenient to exploit the displacement operator $\nabla_{\{D_x^{-1}, Z_0^T\}}(\cdot) : \mathbf{C}^{n\times n} \to \mathbf{C}^{n\times n}$, defined by

$$\nabla_{\{D_x^{-1}, Z_0^T\}}(R) = D_x^{-1} \cdot R - R \cdot Z_0^T, \tag{7.3}$$

which slightly differs from the ones used in [10, 12, 17]. Here $D_x = \mathrm{diag}(x_1, x_2, \cdots, x_n)$ and Z_0 stands for the lower shift matrix. Matching the pattern of definitions in the previous subsection, the matrix $\nabla_{\{D_x^{-1}, Z_0^T\}}(R)$ is called a $\{D_x^{-1}, Z_0^T\}$-*displacement* of R, and the rank of $\nabla_{\{D_x^{-1}, Z_0^T\}}(R)$ is referred to as a $\{D_x^{-1}, Z_0^T\}$-*displacement rank* of R.

It is easy to check that the displacement operator (7.3) transforms a Vandermonde matrix $V(x)$ into a rank-one matrix:

$$\begin{aligned}\nabla_{\{D_x^{-1}, Z_0^T\}}(V(x)) &= D_x^{-1} \cdot V(x) - V(x) \cdot Z_0^T \\ &= \begin{bmatrix} \dfrac{1}{x_1} \\ \dfrac{1}{x_2} \\ \vdots \\ \dfrac{1}{x_n} \end{bmatrix} \cdot [1 \quad 0 \quad \cdots \quad 0],\end{aligned} \tag{7.4}$$

thus justifying the name *Vandermonde-like* for matrices with low $\{D_x^{-1}, Z_0^T\}$-displacement rank.

The following two statements will be used in the next subsection to derive an inversion formula for Vandermonde-like matrices. The first lemma contains an expression for the solution of Vandermonde-like displacement equation.

LEMMA 7.1. *Every matrix $R \in \mathbf{C}^{n\times n}$ is uniquely determined by its $\{D_x^{-1}, Z_0^T\}$-displacement. More precisely, the equality*

$$\begin{aligned}\nabla_{\{D_x^{-1}, Z_0^T\}}(R) &= D_x^{-1} \cdot R - R \cdot Z_0^T \\ &= \sum_{m=1}^{\alpha} \varphi_m \cdot \psi_m^T \qquad (\varphi_m, \psi_m \in \mathbf{C}^n).\end{aligned} \tag{7.5}$$

holds if and only if

$$R = \sum_{m=1}^{\alpha} \mathrm{diag}(D_x \cdot \varphi_m) \cdot V(x) \cdot L(\psi_m)^T, \tag{7.6}$$

where $L(\psi)$ stands for the lower triangular Toeplitz matrix with the first column ψ.

Proof. The spectra of matrices D_x^{-1} and Z_0^T have no intersection, and hence there is only one matrix R satisfying the Sylvester equation (7.5), see e.g. [26].

Let R be given by (7.6), then by using (7.4) we have

$$\begin{aligned}\nabla_{D_x^{-1}, Z_0^T}(R) &= \sum_{m=1}^{\alpha} \operatorname{diag}(D_x \cdot \varphi_m) \cdot \nabla_{\{D_x^{-1}, Z_0^T\}}(V(x)) \\ &\quad \cdot L(\psi_m)^T \\ &= \sum_{m=1}^{\alpha} \operatorname{diag}(D_x \cdot \varphi_m) \cdot \begin{bmatrix} \dfrac{1}{x_1} \\ \dfrac{1}{x_2} \\ \vdots \\ \dfrac{1}{x_n} \end{bmatrix} \\ &\quad \cdot \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \cdot L(\psi_m)^T \\ &= \sum_{m=1}^{\alpha} \varphi_m \cdot \psi_m^T,\end{aligned}$$

and (7.6) follows. ■

The next lemma shows that the $\{D_x^{-1}, Z_0^T\}$-displacement rank of R is essentially inherited by its inverse.

LEMMA 7.2. *Let $\tilde{I}$ is antidiagonal identity matrix, then*

$$\operatorname{rank}\nabla_{\{D_x^{-1}, Z_0^T\}}(R) = \operatorname{rank}\nabla_{\{D_x^{-1}, Z_0^T\}}(R^{-T} \cdot \tilde{I}). \tag{7.7}$$

Proof. Let $\{D_x^{-1}, Z_0^T\}$-displacement rank of R be equal to α, then one can factor (nonuniquely):

$$D_x^{-1} \cdot R - R \cdot Z_0^T = \sum_{m=1}^{\alpha} \varphi_m \cdot \psi_m^T \qquad (\varphi_m, \psi_m \in \mathbf{C}^n).$$

Multiplying the latter equality by R^{-1} from both sides and then taking transposes, one obtains

$$D_x^{-1} \cdot R^{-T} - R^{-T} \cdot Z_0 = \sum_{m=1}^{\alpha} R^{-T} \cdot \psi_m \cdot \varphi_m^T \cdot R^{-T}.$$

Using the identity $Z_0 = \tilde{I} \cdot Z_0^T \tilde{I}$ we further obtain

$$D_x^{-1} \cdot (R^{-T} \cdot \tilde{I}) - (R^{-T} \cdot \tilde{I}) \cdot Z_0^T = \sum_{m=1}^{\alpha} (R^{-T} \cdot \psi_m) \cdot (\varphi_m^T \cdot R^{-T} \cdot \tilde{I}). \tag{7.8}$$

From the latter equality the assertion of Lemma follows. ■

Lemmas 7.1 and 7.2 are the counterparts of the results on Toeplitz-like matrices in [20], which were used by T. Kailath and his coauthors to explain the form of the Gohberg–Semencul formula for inversion of Toeplitz matrices, and to obtain its generalization for the more general Toeplitz-like matrices. A similar explanation for the form of the Gohberg–Olshevsky formulas [11] for inverses of Chebyshev–Vandermonde matrices was given in [24]. In the next subsection we follow the pattern of arguments of [24] to explain the form of the Traub inversion formula (7.1), and to obtain its generalization for Vandermonde-like matrices, thus showing that the latter class also has the property (A).

7.4. *Inversion Formula for Vandermonde-like Matrices*

Let us first justify that the form of (7.1) is a reflection of the displacement structure of $V(x)$. Indeed, using (7.4) one sees from Lemma 7.2 that $\operatorname{rank}(\nabla_{\{D_x^{-1}, Z_0^T\}}(V(x)^{-T} \cdot \tilde{I})) = 1$. Therefore by Lemma 7.1 the matrix $V(x)^{-1}$ must have the form shown in the Traub formula (7.1), implying the following statement.

PROPOSITION 7.3. *The inverse of a Vandermonde matrix is given by*

$$V(x)^{-1} = \begin{bmatrix} d_1 & d_2 & d_3 & \cdots & d_{n-1} & d_n \\ d_2 & d_3 & \cdots & d_{n-1} & d_n & 0 \\ d_3 & & \iddots & \iddots & & \vdots \\ \vdots & \iddots & \iddots & & & \\ d_{n-1} & \iddots & & & & \vdots \\ d_n & 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \cdot V(x)^T \cdot \operatorname{diag}([f_i]_{i=1}^n), \tag{7.9}$$

where

$$V(x) \cdot \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} \dfrac{1}{x_1} \\ \dfrac{1}{x_2} \\ \vdots \\ \dfrac{1}{x_n} \end{bmatrix},$$

$$V(x)^T \cdot D_x^{-1} \cdot \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{7.10}$$

A general inversion formula for Vandermonde-like matrices is given in the next statement, and it is deduced from the formulas (7.8) and (7.6) similarly.

PROPOSITION 7.4. *Let $\{D_x^{-1}, Z_0^T\}$-displacement rank of $R \in \mathbf{C}^{n\times n}$ be equal to α, and let R be specified by its $\{D_x^{-1}, Z_0^T\}$-displacement:*

$$D_x^{-1} \cdot R - R \cdot Z_0^T = \sum_{m=1}^{\alpha} \varphi_m \cdot \psi_m^T \qquad (\varphi_m, \psi_m \in \mathbf{C}^n). \tag{7.11}$$

Then

$$R^{-1} = \sum_{m=1}^{\alpha} \begin{bmatrix} a_1^{(m)} & a_2^{(m)} & a_3^{(m)} & \cdots & a_n^{(m)} \\ a_2^{(m)} & a_3^{(m)} & \cdots & a_n^{(m)} & 0 \\ a_3^{(m)} & & \iddots & 0 & 0 \\ \vdots & \iddots & \iddots & & \vdots \\ a_n^{(m)} & 0 & \cdots & \cdots & 0 \end{bmatrix} \cdot V(x)^T \cdot \operatorname{diag}([c_i^{(m)}]_{1\le i\le n}), \tag{7.12}$$

where $a_m = [a_i^{(m)}]_{1\le i\le n}$, $c_m = [c_i^{(m)}]_{1\le i\le n}$ are determined from the 2α linear systems

$$R \cdot a_m = \varphi_m, \quad R^T \cdot D_x^{-1} C_m = \psi_m, \qquad (m = 1, 2, \cdots, \alpha) \tag{7.13}$$

A different inversion formula for Vandermonde-like matrices was obtained earlier in [17, Theorem II-2.5]. Heinig and Rost did not use their formula to derive an inversion algorithm for Vandermonde-like matrix R. Instead, they gave in [17, Theorem II-2.6] a recursion for the rows of some auxiliary matrix P, then the entries of the inverse matrix are obtained by computing the matrix product $R^{-1} := V(x)^{-1} \cdot P$. The latter procedure of [17] does not seem to be related to the Parker–Traub algorithm, and moreover it is not fast, requiring $O(n^3)$ flops. In Section 8 we use (7.12) to derive fast $O(n^2)$ algorithm for inversion of Vandermonde-like matrices.

7.5. *φ-Cyclic Displacement*

Here we may also remark that in [14] another inversion formula

$$V(x)^{-1} = \begin{bmatrix} \varphi a_1 & \varphi a_2 & \cdots & \varphi a_{n-1} & a_0+\varphi \\ \varphi a_2 & & \iddots & \iddots & a_1 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ \varphi a_{n-1} & \iddots & \iddots & & a_{n-2} \\ a_0+\varphi & a_1 & \cdots & a_{n-2} & a_{n-1} \end{bmatrix}$$
$$\cdot V(x)^T \cdot \operatorname{diag}\left(\left[\frac{1}{P'(x_k)\cdot(\varphi - x_k^n)} \right]_{k=1}^{n} \right), \qquad (7.14)$$

was presented for ordinary Vandermonde matrices. Here φ is arbitrary ($\neq x_k$, $k = 1, 2, \cdots, n$) number, and $P(x)$ and a_k are defined by (1.1). Instead of an upper triangular Hankel matrix as in (7.1), formula (7.14) involves a φ-*circulant* Hankel matrix. This fact can be used to formulate its "displacement" interpretation and generalization, using the so-called φ-cyclic displacement operator

$$\nabla_{\{D_{1/x},\, Z_\varphi^T\}}(R) = D_{1/x}\cdot R - R\cdot Z_\varphi^T,$$

introduced for Vandermonde-like matrices in [12]. Here $Z_\varphi = Z_0 + \varphi e_1 e_n^T$ is the φ-circulant lower shift matrix. The most general inversion formulas for *polynomial Vandermonde-like* matrices appeared in [23].

8. THE GENERALIZED PARKER–TRAUB ALGORITHM

8.1. *Properties* (A)–(D)

In this section we shall show that all the properties (A)–(D) hold not just for Vandermonde matrices, but for Vandermonde-like matrices as well, and shall use this fact to derive a fast $O(n^2)$ *generalized Parker–Traub algorithms* for inversion of Vandermonde-like matrices.

Property (A). Formula (7.12) represents R^{-1} as a sum of α terms, each of which is a product of a Hankel, transposed Vandermonde and a diagonal matrices; this extends the property (A) to Vandermonde-like matrices.

Property (B). The entries of the matrices on the right-hand side of (7.12) are obtained via solving 2α linear systems (7.13) with a Vandermonde-like coefficient matrix R. These systems can be solved in $O(\alpha n^2)$ flops by using a appropriate form of the generalized Schur algorithm [10, 13, 21, 22]; or by many Levinson-type algorithms, see, *e.g.*, [8, 17], to mention just a few. The use of any of these algorithms allows one fast computing the entries of the matrices on the right-hand side of (7.12), and thus extends the property (B) to Vandermonde-like matrices. Among a variety of these possibilities a variant of the generalized Schur algorithm from [10] is the most attractive choice for our purposes here, for the reason, specified in the Property D.

Property (C). The expression on the right-hand side of (7.12) involves α products of a Hankel and a Vandermonde matrix, and each such product can be computed in $O(n^2)$ flops in a similar to (1.6) fashion.

These observations already allow us to devise a fast $O(n^2)$ algorithm for inversion of Vandermonde-like matrices.

Property (D). The algorithm in [10] exploits the fact that R satisfies the displacement equation of the form (7.11) with a *diagonal* matrix D_x, to incorporate partial pivoting technique into fast triangulation procedure for R (without increasing the overall complexity $O(n^2)$ of the algorithm). This allows us to remove the restriction on R to be *a strongly regular* matrix, required by the other algorithms, and moreover, to extend the property (D) to Vandermonde-like matrices.

8.2. *Generalized Parker–Traub Algorithm*

Let a $\{D_x^{-1}, Z_0^T\}$-displacement rank of a Vandermonde-like matrix R be equal to α, and R is given by 2α entries of $\{G_1, B_1\}$ on the right-hand side of

$$\begin{aligned}\nabla_{D_x^{-1}, Z_0^T}(R) &= D_x^{-1} \cdot R - R \cdot Z_0^T \\ &= G_1 \cdot B_1, \qquad G_1, B_1 \in \mathbf{C}^{n\times\alpha}. \end{aligned} \tag{8.1}$$

The following algorithm computes in $O(\alpha n^2)$ flops the entries of the Vandermonde-like matrix R^{-1}, and similarly to (PT1)–(PT2) above, it consists of the following two steps, (G-PT1), (G-PT2).

G-PT1. *Computing the Entries of the Matrices on the Right-Hand Side of* (8.1).

(a) Triangular Factorization of R. Compute in $O(\alpha n^2)$ flops the triangular factorization $R = PLU$ of the permuted version of R, using the variant of [10] of the generalized Schur algorithm, incorporating partial pivoting.

(b) Solving Linear Systems (7.13). This step uses the standard forward and back-substitution technique, and it requires performing $O(\alpha n^2)$ flops.

G-PT2. *Forming the Inverse.* Formula (7.12) represents R^{-1} as the sum of α products of a Hankel, transpose Vandermonde and diagonal matrices. These products can be computed via recursions similar to the one in (1.6). This step requires performing of $O(\alpha n^2)$ flops.

Here is a more detailed formulation of the inversion procedure.

THE GENERALIZED PARKER–TRAUB ALGORITHM.

INPUT: The $\{D_x^{-1}, Z_0^T\}$-displacement $G_1 = [g_{i,j}^{(1)}]_{\substack{1\le i\le n\\ 1\le j\le\alpha}}$, $B_1 - [g_{i,j}^{(1)}]_{\substack{1\le i\le n\\ 1\le j\le\alpha}}$, as on the right hand side of (8.1).

OUTPUT: $R^{-1} = [h_{i,j}]$
COMPLEXITY: $O(\alpha n^2)$ flops.

G-PT1. *Computing the entries of the matrices on the right-hand side of* (8.1).

(a) *Triangular factorization of* R.
(a1) Set $D^{(1)} = \mathrm{diag}(x_1, x_2, \cdots, x_n)$.
(a2) For $k = 1, 2, \cdots, n-1$ do:

(a2.1) Compute

$$[l_{i,k}]_{k \le i \le n} = \sum_{m=1}^{\alpha} b_{k,m}^{(k)} \cdot D^{(k)} \cdot [g_{m,k}^{(k)}]_{k \le m \le n}.$$

(a2.2) Find $l_{q,k} = \max_{k \le i \le n} l_{i,k}$.
Swap x_k and x_q. Swap rows $l_{k,k}$ and $l_{q,k}$.
Swap rows $[g_{k,m}^{(k)}]_{1 \le m \le \alpha}$ and $[g_{q,m}^{(k)}]_{1 \le m \le \alpha}$.
Set P_k be the permutation of the kth and the qth entries.

(a2.3) Compute

$$l_k = [l_{i,k}]_{k \le i \le n} = \frac{1}{l_{k,k}} \cdot [l_{i,k}]_{k \le i \le n},$$

$$u_k = [u_{k,j}]_{k \le j \le n} = x_k \cdot \sum_{m=1}^{\alpha} g_{k,m}^{(k)} \cdot v_m,$$

where $v_m = [v_{m,i}]_{k \le i \le n}$ $(m = 1, 2, \cdots, \alpha)$ are computed by

$$v_{m,k} = b_{m,k}^{(k)},$$

$$v_{m,t} = x_k \cdot v_{m,t-1} + b_{m,t}^{(k)} \qquad (t = k+1, k+2, \cdots, n).$$

(a2.4) Update $G^{(k+1)}$, $B^{(k+1)}$ by

$$\begin{bmatrix} 0 \\ G^{(k+1)} \end{bmatrix} = G^{(k)} - l_k \cdot g^{(k)},$$

$$[0 \quad B^{(k+1)}] = B^{(k)} - b^{(k)} \cdot \frac{u_k}{u_{k,k}},$$

where $g^{(k)} \in \mathbf{C}^{1 \times \alpha}$ and $b^{(k)} \in \mathbf{C}^{\alpha \times 1}$ are the first row of $G^{(k)}$ and the first column of $B^{(k)}$, respectively.

(a3) Set $l_{n,n} = 1$, $u_{n,n} = 1/x_n \cdot \sum_{m=1}^{(n)} g_{n,m}^{(n)} \cdot b_{m,n}^{(n)}$.
(a4) Set $L = [l_{ij}]$, $U = [u_{ij}]$, $P = P_{n-1} \cdot P_{n-2} \cdot \cdots \cdot P_1$.

(b) *Solving linear systems* (7.13). Solve 2α linear systems (7.13) via forward and back-substitution.

G-PT2. *Forming the inverse.* For $j = 1, 2, \cdots, n$ do

G-PT2.1. For $m = 1, 2, \cdots, \alpha$ set $q_m = a^{(m)}$.
G-PT2.2. $h_{n,j} = \sum_{m=1}^{\alpha} (q_m / c_j^{(m)})$.
G-PT2.3. For $i = 1, 2, \cdots, n-1$ do
G-PT2.3.1. $q_m = x_j \cdot q_m + a_{n-i}^{(m)}$.
G-PT2.3.2. $h_{n-i,j} = \sum_{m=1}^{\alpha} (q_m / c_j^{(m)})$.

This algorithm allows us the following conclusion.

PROPOSITION 8.1. *Let a $\{D_x^{-1}, Z_0^T\}$-displacement rank of $R \in \mathbf{C}^{n \times n}$ be equal to α, and let R be given by its $\{D_x^{-1}, Z_0^T\}$-displacement $\{G_1, B_1\}$ on the right-hand side of* (8.1)*. Then the computational complexity of computing the entries of R^{-1} does not exceed $O(\alpha n^2)$ arithmetic operations.*

REFERENCES

1. Björck, A., and Pereyra, V. (1970), Solution of Vandermonde systems of linear equations, *Math. Comput.* **24,** 893–903.
2. Calvetti, D., and Reichel, L. (1993), Fast inversion of Vandermonde-like matrices involving orthogonal polynomials, *BIT* **33,** 473–484.
3. Du Croz, J., and Higham, N. (1992), Stability of methods of matrix inversion, *IMA J. Numer. Ana.* **13,** 1–19.
4. Forney, G. (1966), "Concatenated Codes," M.I.T. Press, Cambridge.
5. Gautschi, W., and Inglese, G. (1988), Lower bounds for the condition number of Vandermonde matrix, *Numer. Math.* **52,** 241–250.
6. Gantmacher, F. R., and Krein, M. G. (1950), "Oscillatory Matrices and Kernels, and Small Vibrations of Mechanical Systems," 2nd ed., Gosudarstvennoe izdatelstvo technicko-teoreticheskoi literatury, Moscow [in Russian].
7. Gohberg, I., and Koltracht, I. (1993), Mixed, componentwise and structured condition numbers, *SIAM J. Matrix Anal. Appl.* **14,** 688–704.
8. Gohberg, I., Kailath, T., Koltracht, I., and Lancaster, P. (1987), Linear complexity parallel algorithms for linear systems of equations with recursive structure, *Linear Algebra Appl.* **88/89,** 271–315.
9. Golub, G., and Van Loan, C. (1989), "Matrix Computations," 2nd ed. Johns Hopkins Univ. Press, Baltimore.

10. Gohberg, I., Kailath, T., and Olshevsky, V. (1995), Fast Gaussian elimination with partial pivoting for matrices with displacement structure, *Math. Comput.* **64,** 1557–1576.
11. Gohberg, I., and Olshevsky, V. (1994), Fast inversion of Chebyshev–Vandermonde matrices, *Numer. Math.* **67** (1), 71–92.
12. Gohberg, I., and Olshevsky, V. (1994), Complexity of multiplication with vectors for structured matrices, *Linear Algebra Appl.* **202,** 163–192.
13. Gohberg, I., and Olshevsky, V. (1994), Fast state space algorithms for matrix Nehari and Nehari–Takagi interpolation problems, *Integral Equations Operator Theory* **20** (1), 44–83.
14. Gohberg, I., and Olshevsky, V. (1994), Fast algorithms with preprocessing for matrix-vector multiplication problems, *Complexity* **10.**
15. Higham, N. (1987), Error analysis of the Björck–Pereyra algorithms for solving Vandermonde systems, *Numer. Math.* **50,** 613–632.
16. Higham, N. (1990), "Stability analysis of algorithms for solving confluent Vandermonde-like systems," *SIAM J. Matrix Anal. Appl.* **11** (1), 23–41.
17. Heinig, G., and Rost, K. (1984), "Algebraic Methods for Toeplitz-like Matrices and Operators," Operator Theory, Vol. 13, Birkauser, Basel.
18. Kaufman, I. (1969), The inversion of the Vandermonde matrix and the transformation to the Jordan canonical form, *IEEE Trans. Automatic Control* **14,** 774–777.
19. Kowalewski, G. (1932), "Interpolation and genäherte Quadratur," Teubner, Berlin.
20. Kailath, T., Kung, S., and Morf, M. (1979), Displacement ranks of matrices and linear equations, *J. Math. Anal. Appl.* **68,** 395–407.
21. Kailath, T., and Sayed, A. H. (1992), Fast algorithms for generalized displacement structures, *in* "Recent Advances in Mathematical Theory of Systems, Control, Networks and Signal Processing II, Proc. of the MTNS-91 Conference" (H. Kimura and S. Kodama, Eds.), pp. 27–32, Mita Press, Japan.
22. Kailath, T., and Sayed, A. H. (1995), Displacement structure: Theory and applications, *SIAM Rev.* **37** (3), 297–386.
23. Kailath, T., and Olshevsky, V. (1994), Displacement structure approach to polynomial Vandermonde and related matrices, *Linear Algebra Appl.*, to appear.
24. Kailath, T., and Olshevsky, V. (1995), Displacement structure approach to Chebyshev–Vandermonde and related matrices, *Integral Equations Operator Theory* **22,** 65–92.
25. Lev-Ari, H., and Kailath, T. (1986), Triangular factorization of structured matrices, *in* "I. Schur Methods in Operator Theory and Signal Processing," Operator Theory: Advances and Applications, Vol. 18 (I. Gohberg, Ed.), Birkhauser, Basel.
26. Lancaster, P., and Tismenetzky, M. (1985), "Theory of Matrices with Applications," 2nd ed., Academic Press, Orlando.
27. Parker, F. (1964), Inverses of Vandermonde matrices, *Amer. Math. Monthly* **71,** 410–411.
28. Reichel, L. (1990), Newton interpolation at Leja points, *BIT* **30,** 23–41.
29. Traub, J. (1966), Associated polynomials and uniform methods for the solution of linear problems, *SIAM Rev.* **8** (3), 277–301.
30. Tyrtyshnikov, E. (1994), How bad are Hankel matrices, *Num. Math.* **67** (2), 261–269.
31. Wertz, H. J. (1965), On the numerical inversion of a recurrent problem: the Vandermonde matrix, *IEEE Trans. Automatic Control* **10** (4), 492.